

# Errata for: A subexponential lower bound for the Random Facet algorithm for Parity Games

Oliver Friedmann\*

Thomas Dueholm Hansen<sup>†</sup>

Uri Zwick<sup>‡</sup>

## Abstract

In Friedmann, Hansen, and Zwick (2011) we claimed that the expected number of pivoting steps performed by the RANDOM-FACET algorithm of Kalai and of Matoušek, Sharir, and Welzl is equal to the expected number of pivoting steps performed by RANDOM-FACET\*, a variant of RANDOM-FACET that bases its random decisions on one random permutation. We then obtained a lower bound on the expected number of pivoting steps performed by RANDOM-FACET\* and claimed that the same lower bound holds also for RANDOM-FACET. Unfortunately, the claim that the expected numbers of steps performed by RANDOM-FACET and RANDOM-FACET\* are the same is false. We provide here simple examples that show that the expected numbers of steps performed by the two algorithms are not the same.

## 1 Introduction

The RANDOM-FACET algorithm was introduced by Kalai [6, 7] and by Matoušek, Sharir, and Welzl [8]. It is a randomized pivoting rule for the simplex algorithm that solves linear programs, and more general *LP-type problems*, in expected time  $2^{O(\sqrt{(m-d)\log d})}$ , where  $m$  is the number of inequalities<sup>1</sup> and  $d$  is the dimension. Kalai's formulation of the algorithm works recursively as follows: Select a uniformly random facet containing the current vertex. Recursively find the optimal solution within the selected facet. If possible, perform an improving pivot from the resulting vertex and repeat. Otherwise return the vertex as the solution.

In Friedmann, Hansen, and Zwick [1] we proved a  $2^{\tilde{\Omega}(\sqrt{m})}$  lower bound for the expected number of pivots performed by a modified variant, RANDOM-FACET\*, of the RANDOM-FACET algorithm. The lower bound was proved for *parity games*, which are of LP-type (see [4]). RANDOM-FACET\* takes as input a fixed permutation of the facets and always selects the first available facet according to this permutation. We claimed that RANDOM-FACET\* performs the same expected number of pivots as RANDOM-FACET when the permutation is chosen uniformly at random. This would imply that our lower bound also holds for the original RANDOM-FACET algorithm. The claim is incorrect, however, and in this errata we explain the error. The main result of [1] is thus flawed.

In Friedmann, Hansen, and Zwick [2] we transferred the lower bound construction from the setting of parity games to the setting of *Markov decision processes*. This proves a lower bound for the RANDOM-

---

\*Department of Computer Science, University of Munich, Germany. E-mail: [Oliver.Friedmann@gmail.com](mailto:Oliver.Friedmann@gmail.com).

<sup>†</sup>Department of Computer Science, Aarhus University, Denmark. E-mail: [tdh@cs.au.dk](mailto:tdh@cs.au.dk).

<sup>‡</sup>School of Computer Science, Tel Aviv University, Israel. E-mail: [zwick@tau.ac.il](mailto:zwick@tau.ac.il).

<sup>1</sup>It is customary to denote the number of inequalities by  $n$ , but we use  $m$  to be consistent with the notation for graphs and shortest paths.

FACET\* algorithm for Markov decision processes, and consequently for linear programming. Again, due to our mistake, the lower bound does not hold for the original RANDOM-FACET algorithm.

It should be stressed that the error is unrelated to the lower bound construction itself; it only involves the relationship between the RANDOM-FACET algorithm and the modified version, RANDOM-FACET\*. In particular, the lower bound for RANDOM-FACET\* remains correct. Also, the main focus of [2] was to prove a lower bound for the RANDOM-EDGE pivoting rule, and this work remains unaffected.

The error was pointed out briefly in [5], and an alternative proof of a  $2^{\tilde{\Omega}(\sqrt[3]{m})}$  lower bound for the RANDOM-FACET algorithm was given. Note that this lower bound is weaker than the  $2^{\tilde{\Omega}(\sqrt{m})}$  lower bound we originally claimed in [1, 2]. The proof in [5] was presented for Markov decision processes. In [3] we transfer the lower bound construction to the setting of shortest paths where we again prove a  $2^{\Omega(\sqrt[3]{m})}$  lower bound for RANDOM-FACET, and a  $2^{\tilde{\Omega}(\sqrt{m})}$  lower bound for RANDOM-FACET\*.

In this errata we explain our error in detail and give examples where the expected running times of RANDOM-FACET and RANDOM-FACET\* differ. We give a short introduction of the algorithms in Section 2. To simplify the presentation we restrict our attention to shortest paths problems. In particular we adopt the notation from [3]. In Section 3 we give an example where RANDOM-FACET\* requires more pivots in expectation than RANDOM-FACET, and in Section 4 we give an example where the opposite is the case. It is not difficult to obtain similar examples for parity games.

## 2 The RANDOM-FACET and RANDOM-FACET\* algorithms

Let  $G = (V, E, c)$  be a weighted directed graph, where  $c : E \rightarrow \mathbb{R}$  is a *cost* function defined on its edges. Let  $\tau \in V$  be a designated *target* vertex. We let  $n = |V \setminus \{\tau\}|$  and  $m = |E|$  be the number of vertices, not counting the target, and edges in  $G$ , respectively. We are interested in finding a tree of shortest paths from all vertices to  $\tau$ .<sup>2</sup>

Let  $B \subseteq E$  be a tree containing directed paths from all vertices to  $\tau$ . For some vertex  $v \in V$ , let  $d_B(v)$  be the distance from  $v$  to  $\tau$  in the tree  $B$ . The RANDOM-FACET algorithm takes two arguments: a set of edges  $F \subseteq E$  and a tree  $B \subseteq F$ . It computes the tree of shortest paths for the subgraph defined by  $F$  as follows. It picks a uniformly random edge  $e \in F \setminus B$ , removes  $e$  from  $F$ , and finds the optimal tree  $B'$  for the resulting subgraph defined by  $F \setminus \{e\}$ . It then checks whether including  $e = (u, v)$  in  $B'$  improves the solution, i.e., whether the path from  $u$  to  $\tau$  that starts with the edge  $e$ , which is currently not in the tree, is shorter than the path from  $u$  to  $\tau$  in the tree. If the path is shorter, then  $e$  is exchanged with the edge  $e'$  emanating from  $u$  in  $B'$ , and a second recursive call is made with  $F$  and  $B'' = B' \cup \{e\} \setminus \{e'\}$ . Note that updating the tree in this way corresponds to an improving pivot of the simplex algorithm. We assume for simplicity that  $G$  does not contain negative cycles. If the path is not shorter, then  $B'$  is a tree of shortest paths for  $F$ .

Pseudo-code for RANDOM-FACET is given on the left of Figure 1. The first argument  $F \subseteq E$  is the set of available edges. Initially  $F = E$ . The second argument  $B$  is the current tree. The call  $\text{IMPROVE}(B', e)$  checks whether  $e$  can be used to improve  $B'$ . The call  $\text{PIVOT}(B', e)$  returns the tree obtained from pivoting with  $e$ , i.e., it exchanges  $e$  with some  $e' \in B'$ .

The RANDOM-FACET\* algorithm is identical to the RANDOM-FACET algorithm, except that it takes as an additional argument a permutation  $\sigma : E \rightarrow \{1, \dots, |E|\}$  of the edges and always removes the first

---

<sup>2</sup>To maintain consistency with [1, 2] it is more convenient for us to work with the *single-target* version of the shortest paths problem, rather than the more standard *single-source* version.

Algorithm RANDOM-FACET( $F, B$ )	Algorithm RANDOM-FACET <sup>*</sup> ( $F, B, \sigma$ )
<pre> <b>if</b> <math>F = B</math> <b>then</b>     <b>return</b> <math>B</math> <b>else</b>     <math>e \leftarrow \text{RANDOM}(F \setminus B)</math>     <math>B' \leftarrow \text{RANDOM-FACET}(F \setminus \{e\}, B)</math>     <b>if</b> <math>\text{IMPROVE}(B', e)</math> <b>then</b>       <math>B'' \leftarrow \text{PIVOT}(B', e)</math>       <b>return</b> <math>\text{RANDOM-FACET}(F, B'')</math>     <b>else</b>       <b>return</b> <math>B'</math> </pre>	<pre> <b>if</b> <math>F = B</math> <b>then</b>     <b>return</b> <math>B</math> <b>else</b>     <math>e \leftarrow \text{argmin}_{e' \in F \setminus B} \sigma(e')</math>     <math>B' \leftarrow \text{RANDOM-FACET}^*(F \setminus \{e\}, B, \sigma)</math>     <b>if</b> <math>\text{IMPROVE}(B', e)</math> <b>then</b>       <math>B'' \leftarrow \text{PIVOT}(B', e)</math>       <b>return</b> <math>\text{RANDOM-FACET}^*(F, B'', \sigma)</math>     <b>else</b>       <b>return</b> <math>B'</math> </pre>

Figure 1: The RANDOM-FACET and RANDOM-FACET<sup>\*</sup> algorithms.

available edge according to this permutation. Pseudo-code for RANDOM-FACET<sup>\*</sup> is shown on the right of Figure 1. Note that RANDOM-FACET<sup>\*</sup> is a deterministic algorithm. We are however interested in the case when the permutation  $\sigma$  is chosen uniformly at random.

Let  $f(F, B)$  be the expected number of pivots performed by RANDOM-FACET( $F, B$ ), and let  $f^*(F, B)$  be the expected number of pivots performed by RANDOM-FACET<sup>\*</sup>( $F, B, \sigma$ ) when  $\sigma$  is uniformly random. In Lemma 4.1 in [1] we claim that  $f(F, B) = f^*(F, B)$  for all  $F \subseteq E$  and all trees  $B \subseteq F$ . This claim is false, and we next show where the mistake was made.

Let  $B$  and  $F$  be given where  $B \subseteq F \subseteq E$ . Suppose  $\sigma$  is a permutation of  $F$  such that the elements of  $F \setminus B$  are ordered uniformly at random. Let  $e = \text{argmin}_{e' \in F \setminus B} \sigma(e')$  be the first available edge from  $F \setminus B$  according to  $\sigma$ . At this stage, selecting  $e$  according to  $\sigma$  is equivalent to selecting  $e \in F \setminus B$  uniformly at random. Let  $B'$  be the optimal tree for  $F \setminus \{e\}$ , and assume that  $e$  improves  $B'$  such that RANDOM-FACET and RANDOM-FACET<sup>\*</sup> perform a second recursive call with  $B'' = \text{PIVOT}(B', e)$ . Note that we may assume that  $B'$  is uniquely determined by  $F$  and  $e$ . In [1] we incorrectly assume that if  $\sigma$  is uniformly random then the elements of  $F \setminus B''$  are also ordered uniformly at random, and it is this assumption that is the source of our error.

In Sections 3 and 4 we give two concrete examples in which  $f(F, B) < f^*(F, B)$  and  $f(F, B) > f^*(F, B)$ , respectively. The examples are for shortest paths. In the remainder of this section we consider an abstract example that illustrates why the permutation is not uniformly random for the second recursive call.

Suppose  $F = \{1, 2, 3\}$  and  $B = \{1\}$ . For some permutation  $\sigma$ , we write  $a \prec b$  as shorthand notation for  $\sigma(a) < \sigma(b)$ . There are six permutations of  $F$ :

$$\begin{array}{lll}
 1 \prec 2 \prec 3 & 2 \prec 1 \prec 3 & 3 \prec 1 \prec 2 \\
 1 \prec 3 \prec 2 & 2 \prec 3 \prec 1 & 3 \prec 2 \prec 1
 \end{array}$$

As there is an equal number of permutations in which  $2 \prec 3$  and  $3 \prec 2$ , the first random choice made by RANDOM-FACET<sup>\*</sup> is uniform. However, if, for example,  $2 \prec 3$ , then only three equally likely permutations remain:

$$\begin{array}{lll}
 1 \prec 2 \prec 3 & 2 \prec 1 \prec 3 & 2 \prec 3 \prec 1
 \end{array}$$

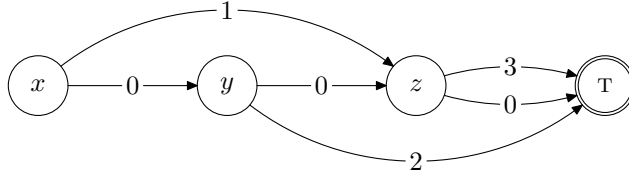


Figure 2: A shortest paths problem in which the expected number of improving switches performed by RANDOM-FACET and RANDOM-FACET\* differ.

Now, if  $B'' = \{2\}$ , then the distribution of the elements from  $F \setminus B'' = \{1, 3\}$  is no longer uniform for the second recursive call of RANDOM-FACET\*. We now have  $1 \prec 3$  with probability  $2/3$ , and  $3 \prec 1$  with probability  $1/3$ . Thus, elements that leave  $B$  are more likely to be picked sooner in the second recursive call.

### 3 Example: RANDOM-FACET\* can be slower than RANDOM-FACET

Consider the simple example of a shortest paths problem given in Figure 2. There are three non-terminal vertices  $x, y, z$ , each with two out-going edges. We refer to the edges leaving  $x$  as  $x_0$  and  $x_1$ , where the cost of  $x_0$  is 0 and the cost of  $x_1$  is positive. We refer to the other edges in a similar fashion such that the set of edges is  $E = \{x_0, x_1, y_0, y_1, z_0, z_1\}$ . Note that a tree  $B = \{x_i, y_j, z_k\} \subseteq E$  can be viewed as a binary string  $ijk$  of length 3. In particular the set of trees can be viewed as vertices of a cube, and in fact the corresponding linear program is combinatorially equivalent to a cube.

Consider two neighboring vertices of the cube. Moving from one vertex to the other corresponds to performing a pivot step, and we orient the edge in the improving direction. When all edges are oriented, the resulting orientation is known as an *acyclic unique sink orientation* (see, e.g., [9]). Note that a set of edges  $F$  defines a corresponding *face* that contains the trees in  $F$ . In every such face there is a unique optimal solution, which means that in the subcube there is a unique vertex where all the edges are incoming. Such a vertex is called a *sink*.

The orientation corresponding to the graph in Figure 2 is shown in Figure 3. For the example we start with the tree  $B = \{x_0, y_0, z_1\}$ , which corresponds to the vertex 001. The optimal solution is the vertex 000, and there are three paths (sequences of pivots) leading from 001 to 000. These paths are shown at the top of Figure 3. The vertex 000 is in the lower left corner of the cube, and the cube is oriented according to the axes shown on the left.

The bottom of Figure 3 shows the computations performed by RANDOM-FACET and RANDOM-FACET\*. The set  $F = \{x_0, x_1, y_0, y_1, z_0, z_1\}$  is the set of edges given as an argument to the calls  $\text{RANDOM-FACET}(F, B)$  and  $\text{RANDOM-FACET}^*(F, B, \sigma)$ . The elements in  $B$  are underlined, and  $\sigma$  is assumed to be uniformly random. The labelled squares indicate the edges that are chosen by the algorithms. The probabilities of picking the edges are shown as well. In most cases the probabilities are the same, however, the label  $\frac{1}{2}$  vs  $\frac{3}{8}$  means that RANDOM-FACET picks the edge with probability  $\frac{1}{2}$  and RANDOM-FACET\* picks the edge with probability  $\frac{3}{8}$ . The lines below the squares indicate recursive calls with the chosen edge. The first recursive call is shown on the left, and if a second recursive call is made, it is shown on the right. Lines corresponding to second recursive calls are labelled by the pivot that is made. Recall that the vertex returned by the first recursive call is uniquely determined. When only a single sequence of pivots can be generated in a subtree this sequence is shown instead of the subtree itself. It

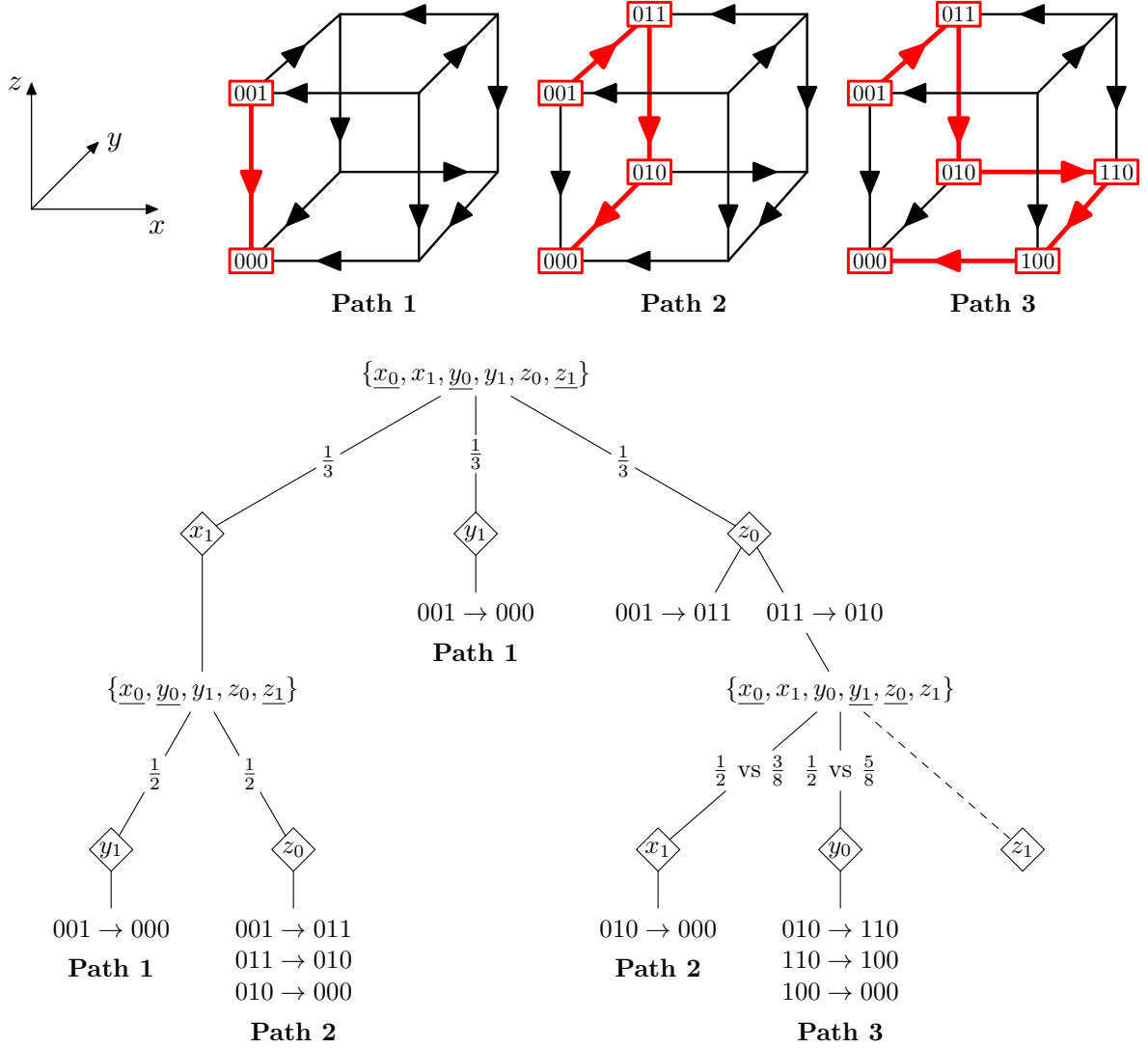


Figure 3: Paths generated by the RANDOM-FACET and RANDOM-FACET\* algorithms when starting from 001.

is also shown which of the paths the sequence corresponds to. For a selection of edges, the sequence of pivots for the generated path can be read in a pre-order traversal of the tree.

The line to  $z_1$  is dashed because removing this edge does not affect the behavior of the algorithms. More precisely there is no path from 010 that leads back to the  $z_1$  facet. (In fact, the edge that leaves a tree during a pivot can in general never appear in the corresponding second recursive call, and it may be viewed as if it has been removed.) We instead combine the probabilities that should have appeared after  $z_1$  with those at the same recursive call as  $z_1$ .

Consider the path in the computation tree that first picks  $z_0$  and then picks  $y_0$ . In order for the call

$$\text{RANDOM-FACET}^*(\{x_0, x_1, y_0, y_1, z_0, z_1\}, \{x_0, y_0, z_1\}, \sigma)$$

to realize this path, the permutation  $\sigma$  must satisfy  $z_0 \prec x_1$ ,  $z_0 \prec y_1$ , and  $y_0 \prec x_1$ . There are 150 permutations that satisfy these restrictions, which means that the path is generated with probability

$150/6! = 5/24$ , and the edge is picked with probability  $5/8$ , given that the call is made. The permutations can be obtained by adding  $x_0$  and  $z_1$  to the following five permutations:

$$\begin{array}{lll} z_0 \prec z_0 \prec x_1 \prec y_1 & z_0 \prec y_0 \prec x_1 \prec y_1 & z_0 \prec y_1 \prec y_0 \prec x_1 \\ z_0 \prec z_0 \prec y_1 \prec x_1 & z_0 \prec y_0 \prec y_1 \prec x_1 & \end{array}$$

The same choice is made with probability  $1/2$  by RANDOM-FACET. Note that  $y_0$  was part of the original tree  $B$ , and as illustrated in Section 2 this means that it is more likely to be picked during a second recursive call, which is exactly the behavior we observe here. The resulting path, **Path 3**, is longer than the path, **Path 2**, generated when  $x_1$  is picked. Thus the expected running time is worse for RANDOM-FACET\* than for RANDOM-FACET. To be exact, the expected number of pivots made by RANDOM-FACET is

$$f(E, \{x_0, y_0, z_1\}) = \frac{1}{3} \left( \frac{1}{2} + \frac{3}{2} \right) + \frac{1}{3} + \frac{1}{3} \left( \frac{3}{2} + \frac{5}{2} \right) = \frac{7}{3} \approx 2.333 ,$$

and the expected number of pivots made by RANDOM-FACET\* is

$$f^*(E, \{x_0, y_0, z_1\}) = \frac{1}{3} \left( \frac{1}{2} + \frac{3}{2} \right) + \frac{1}{3} + \frac{1}{3} \left( \frac{9}{8} + \frac{25}{8} \right) = \frac{29}{12} \approx 2.417 .$$

#### 4 Example: RANDOM-FACET\* can be faster than RANDOM-FACET

We next give an example for which RANDOM-FACET\* is faster than RANDOM-FACET. In fact the example is again for the graph shown in Figure 2, and we therefore use the same notation as in Section 3. The only difference is that we now start from the vertex 111. The resulting paths and computation tree are shown in Figure 4. There are again only three paths from 111 to 000.

Consider the path in the computation tree that first picks  $z_0$  and then picks  $x_1$ . Note that  $x_1$  is part of the original tree  $B = \{x_1, y_1, z_1\}$ , and we would therefore expect this choice to be made with higher probability for RANDOM-FACET\* than for RANDOM-FACET, which is exactly the case. In order for the call

$$\text{RANDOM-FACET}^*(\{x_0, x_1, y_0, y_1, z_0, z_1\}, \{x_1, y_1, z_1\}, \sigma)$$

to realize the path, the permutation  $\sigma$  must satisfy  $z_0 \prec x_0$ ,  $z_0 \prec y_0$ , and  $x_1 \prec y_0$ . There are 150 permutations that satisfy these restrictions, which means that the path is generated with probability  $150/6! = 5/24$ , and the edge is picked with probability  $5/8$ , given that the call is made. The permutations can be obtained by adding  $y_1$  and  $z_1$  to the following five permutations:

$$\begin{array}{lll} x_1 \prec z_0 \prec x_0 \prec y_0 & z_0 \prec x_1 \prec x_0 \prec y_0 & z_0 \prec x_0 \prec x_1 \prec y_0 \\ x_1 \prec z_0 \prec y_0 \prec x_0 & z_0 \prec x_1 \prec y_0 \prec x_0 & \end{array}$$

The same choice is again made with probability  $1/2$  by RANDOM-FACET. This time, the resulting path, **Path 2**, is shorter than the path, **Path 3**, generated when  $y_0$  is picked. Thus the expected running time is better for RANDOM-FACET\* than for RANDOM-FACET. To be exact, the expected number of pivots made by RANDOM-FACET is

$$f(E, \{x_0, y_0, z_1\}) = 1 + \frac{1}{3} \left( \frac{3}{2} + \frac{5}{2} \right) + \frac{1}{3} \left( \frac{3}{2} + \frac{5}{2} \right) = \frac{11}{3} \approx 3.667 ,$$

and the expected number of pivots made by RANDOM-FACET\* is

$$f^*(E, \{x_0, y_0, z_1\}) = 1 + \frac{1}{3} \left( \frac{3}{2} + \frac{5}{2} \right) + \frac{1}{3} \left( \frac{15}{8} + \frac{15}{8} \right) = \frac{43}{12} \approx 3.583 .$$

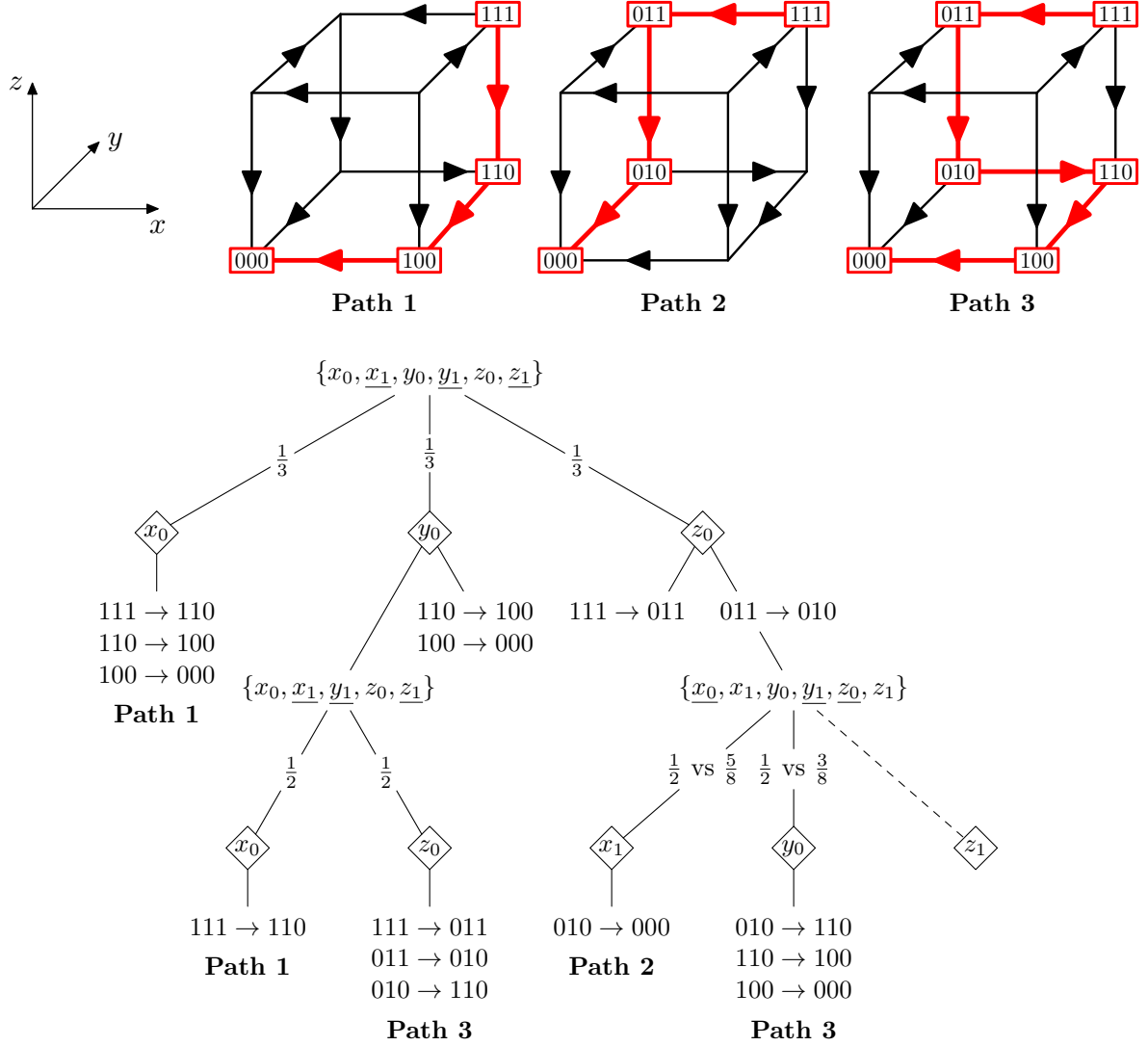


Figure 4: Paths generated by the RANDOM-FACET and RANDOM-FACET\* algorithms when starting from 111.

## Acknowledgement

We would like to thank Bernd Gärtner, Günter Rote and Tibor Szabó for various discussions on the RANDOM-FACET algorithm and its variants that helped us realize that the expected number of pivoting steps performed by RANDOM-FACET and RANDOM-FACET\* are *not* the same.

## References

- [1] O. Friedmann, T. D. Hansen, and U. Zwick. A subexponential lower bound for the random facet algorithm for parity games. In *Proc. of 22nd SODA*, pages 202–216, 2011.
- [2] O. Friedmann, T. D. Hansen, and U. Zwick. Subexponential lower bounds for randomized pivoting rules for the simplex algorithm. In *Proc. of 43th STOC*, pages 283–292, 2011.

- [3] O. Friedmann, T. D. Hansen, and U. Zwick. Random-Facet and Random-Bland require subexponential time even for shortest paths. *CoRR*, abs/1410.7530, 2014.
- [4] N. Halman. Simple stochastic games, parity games, mean payoff games and discounted payoff games are all LP-type problems. *Algorithmica*, 49(1):37–50, 2007.
- [5] T. D. Hansen. *Worst-case Analysis of Strategy Iteration and the Simplex Method*. PhD thesis, Aarhus University, 2012. Available at: [www.cs.au.dk/~tdh/papers/dissertation.pdf](http://www.cs.au.dk/~tdh/papers/dissertation.pdf).
- [6] G. Kalai. A subexponential randomized simplex algorithm (extended abstract). In *Proc. of 24th STOC*, pages 475–482, 1992.
- [7] G. Kalai. Linear programming, the simplex algorithm and simple polytopes. *Mathematical Programming*, 79:217–233, 1997.
- [8] J. Matoušek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. *Algorithmica*, 16(4-5):498–516, 1996.
- [9] T. Szabó and E. Welzl. Unique sink orientations of cubes. In *Proc. of 42th FOCS*, pages 547–555, 2001.